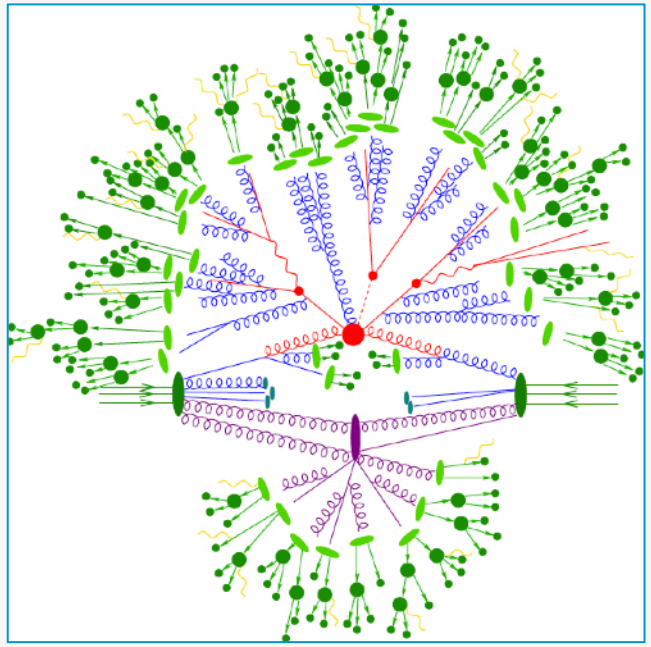
Event Generators

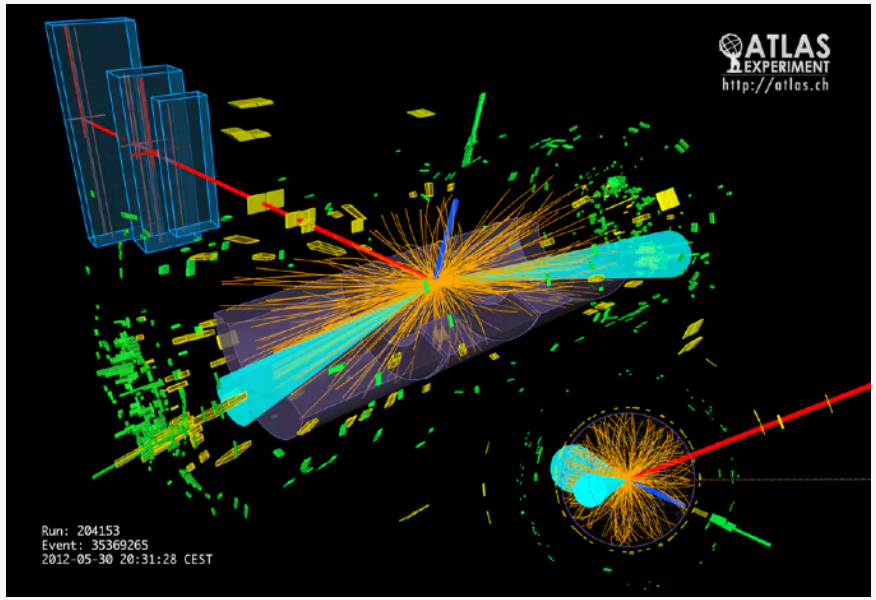
HEP-CCE All-hands Update

J. Taylor Childers (Argonne)
Stefan Höche (Fermilab)

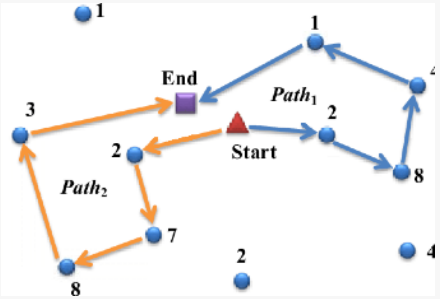
Summary

- Event Generators in the LHC provide a unique opportunity to leverage HPCs.
  - They are experiment independent and compute intense.
- Primary NLO generators for the LHC are MadGraph and Sherpa.
- Primary LO generator for the LHC is largely Pythia8
- Challenges include:
  - Generator development has historically not been supported by the DOE, leading to most generator teams being based outside the US. This is changing.
  - Not easy for a software engineer to pick up and just re-factor for performance. Highly desktop-CPU optimized algorithms resulted in obscure algorithmic choices.
- But we progress.





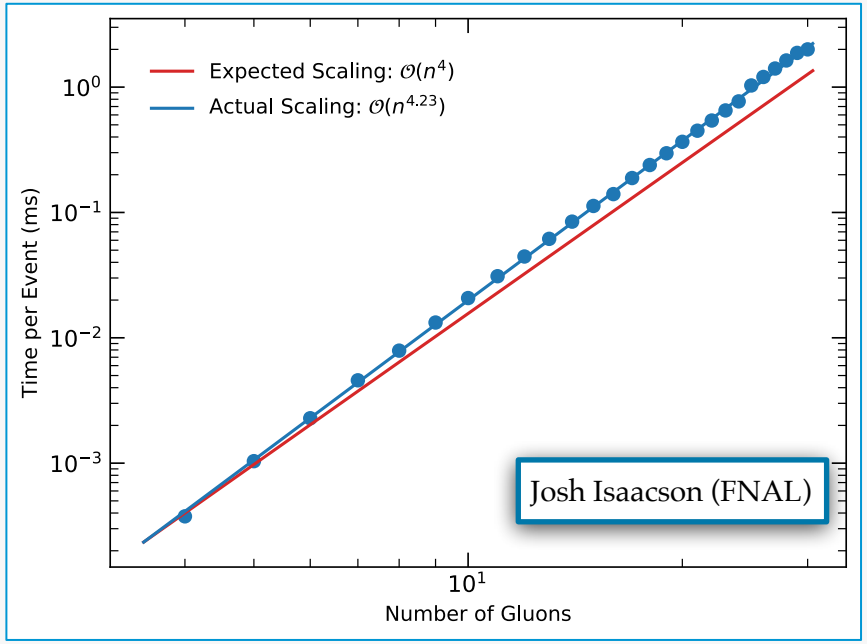
Two Groups, Two Approaches



- MadGraph5: CERN: Stefan Roiser, Andrea Valassi, Laurence Field, Olivier Mattelaer, ANL: Walter Hopkins, Tyler Burch, Taylor Childers, Smita Darmora
  - MG auto-generates compilable code based on what the user wants.
  - Generated a few of these processes, converted them to CUDA and Kokkos
  - MG devs working to reverse engineer the conversions back into the auto-generation code.
- Sherpa: FNAL: Stefan Höche, Steve Mrenna, Josh Isaacson, LUND: Stefan Prestel, UCinn: Holger Schulz
  - Taking a more ground up approach
  - Re-engineering algorithms for parallelization at many scales

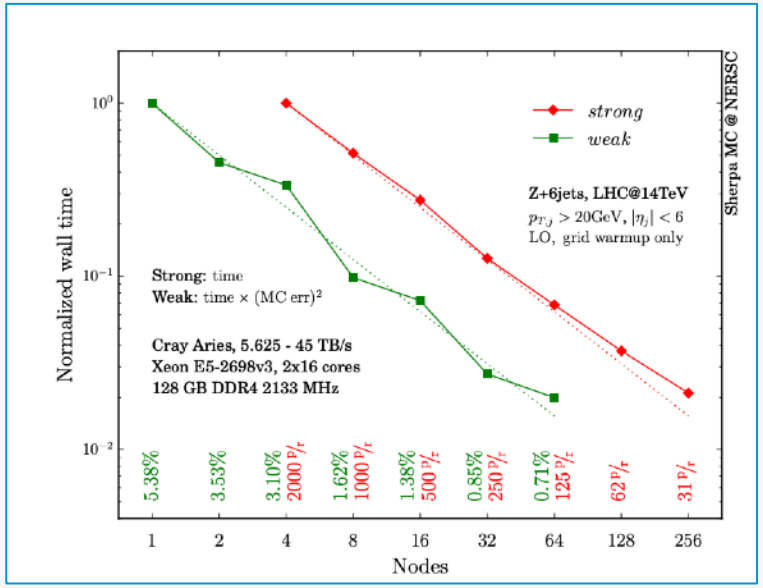
Ground Up Improvements

- Sherpa team working on re-engineering high-multiplicity algorithms that tend to lead compute intensive LHC calcs
- This work focuses on making a key calculation GPU friendly.
- The Berends-Giele algorithm for color-ordered amplitudes (no color factors included yet) is shown here at varying gluons.
- GPU used: Telsa P100 16 GB
- Each thread calculates one event (not many threads per event) => 1024 events at a time
- Currently, memory limits the maximum number of events to be stored on the GPU at a given time.



Ground Up Improvements

- Additional work to re-engineering the event generation workflow to be more scalable.
- Published in 2019 was some initial work on LO boson production at scale.
- This work revisited the event generation workflow (integration, generate, unweight,) to scale up on current HPC systems.
- Used Comix (part of Sherpa) and Pythia.
- This work continues with NLO calculations.



https://arxiv.org/pdf/1905.05120.pdf

MadGraph GPU Porting Effort

- The MadGraph team has focused on how to convert their framework to auto-generate CUDA algorithms.
- These have largely, though not entirely, been based on line-by-line conversion and not re-engineering algorithms for GPUs.
- The GitHub repo contains autogenerated code for a few processes, which have been converted to CUDA and now Kokkos.
- These examples are being used by the MG developers (Olivier) to convert the code-generation to output CUDA (not yet Kokkos).
- Currently this is all based on a simple Rambo phase-space generation
- We don't have any detailed results just yet, besides code.

https://github.com/madgraph5/madgraph4gpu

```
11
12  #include <cmath>
13  #include <thrust/complex.h>
14
15
16  using namespace std;
17
18  namespace MG5_sm
19  {
20  __device__ void oxxxxx(const double p[3], double fmass, int nhel, int nsf,
21  thrust::complex<double> fo[6]);
22
23  __device__ void sxxxxx(const double p[3], int nss, thrust::complex<double> sc[3]
24
25  __device__ void ixxxxx(const double p[3], double fmass, int nhel, int nsf,
26  thrust::complex<double> fi[6]);
27
28  __device__ void txxxxx(const double p[3], double tmass, int nhel, int nst,
29  thrust::complex<double> fi[18]);
30
31  __device__ void vxxxxx(const double p[3], double vmass, int nhel, int nsv,
32  thrust::complex<double> v[6]);
33
```

Machine Learning for EvGen

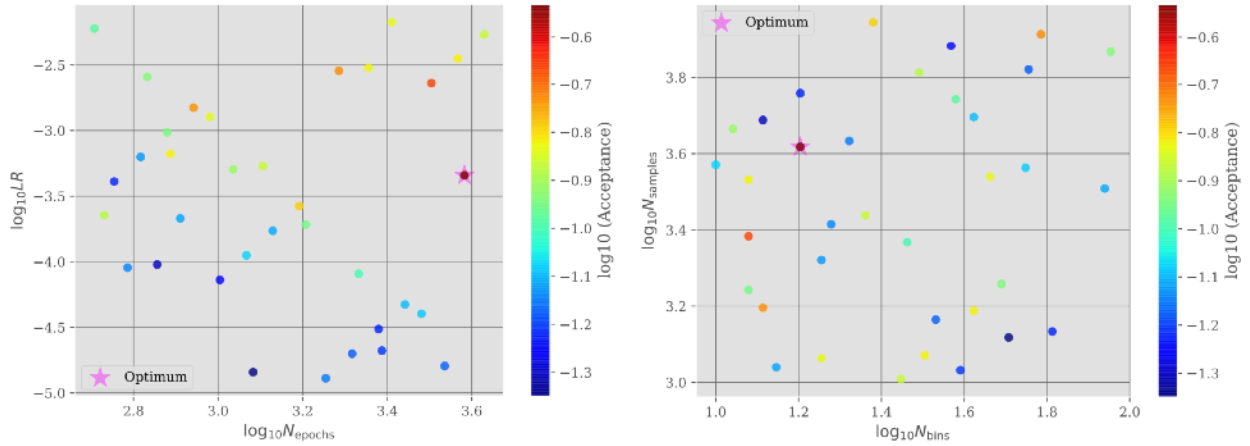


FIG. 2: Projections of the sampled parameters and color coded acceptances. The plot on the left suggest a high learning rate, coupled with a large number of epochs to be beneficial. The plot on the right suggests a strong preference for a small number of bins. The best performing configuration is indicated with a star.

unweighting efficiency $\langle w\rangle/w_{\max}$		LO QCD $n=0$	LO QCD $n=1$	LO QCD $n=2$	LO QCD $n=3$	LO QCD $n=4$	NLO QCD (RS) $n=0$	NLO QCD (RS) $n=1$
$W^+ + n$ jets	Sherpa	$2.8 \cdot 10^{-1}$	$3.8 \cdot 10^{-2}$	$7.5 \cdot 10^{-3}$	$1.5 \cdot 10^{-3}$	$8.3 \cdot 10^{-4}$	$9.5 \cdot 10^{-2}$	$4.5 \cdot 10^{-3}$
	NN+NF	$6.1 \cdot 10^{-1}$	$1.2 \cdot 10^{-1}$	$1.0 \cdot 10^{-2}$	$1.8 \cdot 10^{-3}$	$8.9 \cdot 10^{-4}$	$1.6 \cdot 10^{-1}$	$4.1 \cdot 10^{-3}$
	Gain	2.2	3.3	1.4	1.2	1.1	1.6	0.91
$W^- + n$ jets	Sherpa	$2.9 \cdot 10^{-1}$	$4.0 \cdot 10^{-2}$	$7.7 \cdot 10^{-3}$	$2.0 \cdot 10^{-3}$	$9.7 \cdot 10^{-4}$	$1.0 \cdot 10^{-1}$	$4.5 \cdot 10^{-3}$
	NN+NF	$7.0 \cdot 10^{-1}$	$1.5 \cdot 10^{-1}$	$1.1 \cdot 10^{-2}$	$2.2 \cdot 10^{-3}$	$7.9 \cdot 10^{-4}$	$1.5 \cdot 10^{-1}$	$4.2 \cdot 10^{-3}$
	Gain	2.4	3.3	1.4	1.1	0.82	1.5	0.91
$Z + n$ jets	Sherpa	$3.1 \cdot 10^{-1}$	$3.6 \cdot 10^{-2}$	$1.5 \cdot 10^{-2}$	$4.7 \cdot 10^{-3}$		$1.2 \cdot 10^{-1}$	$5.3 \cdot 10^{-3}$
	NN+NF	$3.8 \cdot 10^{-1}$	$1.0 \cdot 10^{-1}$	$1.4 \cdot 10^{-2}$	$2.4 \cdot 10^{-3}$		$1.8 \cdot 10^{-3}$	$5.7 \cdot 10^{-3}$
	Gain	1.2	2.9	0.91	0.51		1.5	1.1

TABLE II: Unweighting efficiencies at the LHC at $\sqrt{s} = 14$ TeV using the NNPDF 3.0 NNLO PDF set and a correspondingly defined strong coupling. Jets are identified using the k_T clustering algorithm with $R = 0.4$, $p_{T,j} > 20$ GeV and $|\eta_j| < 6$. In the case of Z/γ^* production, we also apply the invariant mass cut $66 < m_{ll} < 116$GeV.

https://arxiv.org/pdf/2001.10028.pdf

- Sherpa group has also investigated using Neural Networks to guide phase space integration, as a VEGAS replacement.
- NNs can be GPU friendly and thus moving from VEGAS to NNs could improve the usefulness of HPCs for EvGen.
- "The new integrator based on Neural Networks and Normalizing Flows gives a much larger unweighting efficiency than Sherpa in processes with few jets, both at LO and at NLO precision. In processes with more final-state jets it performs similarly to the existing integration techniques in Sherpa. […] We expect that in the high multiplicity cases the Neural Network + Normalizing Flow technique will also outperform Sherpa, if it can be trained over sufficiently many epochs with sufficiently many sample points."

Summary

- EvGen and HEP software communities are working to make these tools utilize accelerators and scale up on HPCs.
- Given their independence from experimental software frameworks and concentration of computation, they are easier targets for optimization.
- Moving these frameworks forward requires a lot of expert involvement due to the complex algorithms that have been developed.
- MadGraph team working on establishing some metrics of success, including physics validation. Hope to have some results and comparisons between CPU, CUDA, Kokkos in coming months.
- Work on building new accelerator friendly EvGen algorithms and workflows should be providing new results soon. Publications to follow.